APPLICATION

FOR

UNITED STATES LETTERS PATENT

TITLE: OBTAINING CYCLIC REDUNDANCY CODE

APPLICANT: STEVE H. WEISSINGER

CERTIFICATE OF MAILING BY EXPRESS MAIL

Express Mail Label No. EL870691485US

I hereby certify that this correspondence is being deposited with the United States Postal Service as Express Mail Post Office to Addressee with sufficient postage on the date indicated below and is addressed to the Commissioner for Patents, Washington, D.C. 20231.

2-15-02
Date of Deposit

Signature

Gabriel Lewis
Typed or Printed Name of Person Signing Certificate

Attorney's Docket No.: 10559-576001
Client's Ref. No.: P12790

COMBINED DECLARATION AND POWER OF ATTORNEY

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name.

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled OBTAINING CYCLIC REDUNDANCY CODE, the specification of which:

[X] is attached hereto.
[] was filed on _ as Application Serial No. _ and was amended on ________________.
[] was described and claimed in PCT International Application No. ________________ filed on ________________ and as amended under PCT Article 19 on ________________.

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose all information I know to be material to patentability in accordance with Title 37, Code of Federal Regulations, §1.56.

I hereby appoint the following attorneys and/or agents to prosecute this application and to transact all business in the Patent and Trademark Office connected therewith:

Scott C. Harris, Reg. No. 32,030
Ralph A. Mittelberger, Reg. No. 33,195
John R. Wetherell, Jr., Reg. No. 31,678
Paul A. Pysher, Reg. No. 40,780
Frederick H. Rabin, Reg. No. 24,488
Joseph R. Baker, Reg. No. 40,900
Samuel Borodach, Reg. No. 38,388
Christopher Centurelli Reg. No. 44,599
David L. Feigenbaum, Reg. No. 30,378
Hans R. Troesch, Reg. No. 36,950
Bing Ai, Reg. No. 43,312
Samuel L. Lee, Reg. No. 42,791
John C. Phillips, Reg. No. 35,322
Richard J. Anderson, Reg. No. 36,732
William J. Egan, III, Reg. No. 28,411

Address all telephone calls to Scott C. Harris at telephone number (858) 678-5070.

Address all correspondence to SCOTT C. HARRIS at:

FISH & RICHARDSON P.C.
4350 La Jolla Village Drive, Suite 500
San Diego, CA 92122

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patents issued thereon.

Attorney's Docket No.: 10559-576001
Client's Ref. No.: P12790

Combined Declaration and Power of Attorney

Page 2 of 2 Pages

Full Name of Inventor: STEVE H. WEISSINGER

Inventor's Signature: Steve Weissinger Date: FEB 4 2002

Residence Address: 641 Brenda Lee Drive, San Jose, CA 95123

Citizenship: U.S.A.

Post Office Address: 641 Brenda Lee Drive, San Jose, CA 95123

20382156.doc

OBTAINING CYCLIC REDUNDANCY CODE

TECHNICAL FIELD

This application relates to verifying the integrity of data transmissions, and more particularly, to verifying the integrity of digital data transmissions using cyclic redundancy codes.

BACKGROUND

Data transmission 10 (FIG. 1) involves a transfer of information known as data between a sender 12 and a receiver 14. Often, data transmission 10 includes information transmitted as digital bits of ones and zeros, represented here by m_{n-1} to m_0, and referred to as a message M.

In a perfect world, message M transmits free of errors. Unfortunately, errors are often introduced though medium 16 by which message M travels from sender 12 to receiver 14 (e.g., medium 16 may be any combination of wire, cable, fiber-optic, air, or link layer devices). One method for detecting the presence of errors in message M employs cyclic redundancy codes.

Cyclic redundancy codes treat groupings of digital bits like message M as a polynomial where each bit in the grouping represents a coefficient in the polynomial $X^{n-1} + X^{n-2} + X^0$. For example, a group of eight bits 11001101 may be represented by polynomial $X^7 + X^6 + X^3 + X^2 + 1$ (i.e., $1*X^7 + 1*X^6 + 0*X^5 + 0*X^4 + 1*X^3 + 1*X^2 + 0*X^1 + 1*X^0$).

These polynomials form an algebraic object known as a commutative ring with coefficients in Z/p where Z are the integers and p is a prime number, here 2, also known as {0,1} modulo 2. A non empty set R together with two binary operations {+*} is called a ring if (R+) is an abelian group, (R*) is a

semi group and the distributive laws are obeyed, (i.e. a*(b+c) = a*b+a*b).

In polynomial rings, there are no carries or borrows from one coefficient to the next. In arithmetic modulo 2, addition and subtraction are identical and may be implemented with exclusive-or. For example:

```
 10111011    10110011    11110000    00010101
+11001110   +11000101   -00100110   -10101111
 01110101    01110110    11010110    10111010
```

Division of polynomials represented as groups of bits is completed in a manner similar to binary division except subtraction is done in modulo2. A divisor will 'go into' a dividend if the dividend polynomial is of the same degree as the divisor polynomial (i.e., the divisor and dividend share at least the same most significant bit).

A cyclic redundancy code may be obtained by calculating the remainder for message M divided by a generator polynomial P. This remainder is called a cyclic redundancy code ("CRC").

To obtain a CRC for a message M, the group of bits to be divided by generator polynomial P may include appended zero-bits 17. Zero-bits 17 are equal in number to the degree of generator polynomial P. Thus, the CRC of a message M = 10111*000* having three appended zero-bits 17 based on a generator polynomial P = X^3+1 = 1001 of degree three (i.e., where X^3 is the most significant bit of polynomial P) may be calculated as follows:

```
          00010101
    1001 |10111000  <---  M including 3 appended zero
          1001            bits for Deg(P)=3
  P        0101
           0000
            1010
            1001
             0110
             0000
              1100
```

```
remainder    1001
              101 = M(modulo)P = MOD(M,P) = CRC.
```

The resulting remainder, shown as CRC 18, may be appended to message M, replacing zero bits 17, to create a message M′. Sender 12 transmits message M′ via medium 16 to receiver 14 as data transmission 10.

Upon receipt, receiver 14 divides message M′ by the same generator polynomial P to obtain a CRC for M′ and check the validity of data transmission 10. If the resulting remainder is zero (i.e., CRC = M′(modulo)P = 0), the integrity of data transmission 10 is confirmed. For example:

```
               00010101
         1001 |10111101  <---  M′ with CRC = 101 appended
               1001
  P             0101
                0000
                 1011
                 1001
                  0100
                  0000
                   1001
remainder          1001
                    000 = M′(modulo)P = 0 = valid data.
```

If the remainder of message M′ divided by polynomial P is not zero (i.e., CRC = M′(modulo)P ≠ 0), data transmission 10 contains one or more errors. For example:

```
               00010110
         1001 |10101101  <---  M′ with error 0 introduced
               1001            to message M′
  P             0111
                0000
                 1111
                 1001
                  1100
                  1001
                   1011
remainder          1001
                    010 = M′(modulo)P ≠ 0 = error.
```

DESCRIPTION OF DRAWINGS

FIG. 1 shows a prior art block diagram of a data transmission employing cyclic redundancy codes.

FIG. 2 shows a process for obtaining a CRC where a message M is separated into a plurality of segments.

FIG. 3 shows a CRC generator for obtaining a CRC of a message according to the process in FIG. 2.

FIG. 4 shows the CRC generator of FIG. 3 separated into four segments.

FIG. 5 shows a CRC generator for obtaining a CRC of a message M according to the process in FIG. 2.

FIG. 6 shows a modulo unit for obtaining a remainder of a message using a reciprocal approximation of a generator polynomial.

FIG. 7 shows the CRC generator in FIG. 3 using the modulo unit in FIG. 6.

FIG. 8 shows a CRC generator for updating a CRC of a message M where a portion of message M has been adjusted.

FIG. 9 shows the CRC generator in FIG. 8 operating on an adjusted segment of message M.

FIG. 10 is a view of computer hardware used to implement an embodiment of the invention.

Like reference symbols in the various drawings indicate like elements.

DETAILED DESCRIPTION

Process 20 (FIG. 2) obtains a CRC for a message M based on a generator polynomial P.

Process 20 includes separating (201) a message M into a plurality of message segments M_s; moduloing (defined below) (203) the message segments by a generator polynomial P, if needed, to

obtain a remainder R for each segment; multiplying (205) the remainder R for each segment by an appropriate segment-constant C to obtain a segment-remainder SR for each segment M_s; accumulating (207) the segment-remainders SR for each segment M_s to obtain an accumulated-remainder AR for message M; and moduloing (209) the accumulated-remainder by generator polynomial P, if needed, to obtain the CRC for message M.

Separating (201) message M into a plurality of message segments M_s includes parsing message M so that:

$$M = M_{s-1}*X^{n*(s-1)} + M_{s-2}*X^{n*(s-2)}\ldots + M_1*X^{n(1)} + M_0*X^{n(0)};$$

where s is the number of segments into which message M is separated, n is the number of bits in each segment, X is the position of each segment in message M, and M_s are the individual segments of message M. If the number of bits is n, then X is of the form X=[1000…0] where there are n zeroes, n+1 elements, and X is of degree n. Multiplying M by X will shift the message left by n bits. Multiplying M by X^2 will shift the message by 2n bits (and so on).

Moduloing (203) includes obtaining a remainder R for each message segment M_s by dividing segment M_s by generator polynomial P if the degree of the most significant bit of segment M_s is the same as or greater than the degree of the most significant bit of polynomial P. If the degree of segment M_s is less than the degree of polynomial P (i.e., where the most significant bit of M_s is smaller than the most significant bit of polynomial P) moduloing (203) is not needed since the remainder for message segment M_s equals segment M_s itself. In alternate embodiments moduloing (203) may be accomplished by multiplying message segment M_s by a reciprocal approximation for polynomial P, rather than dividing segment M_s by polynomial P, to obtain remainder R for message segment M_s. The operation of multiplication by

reciprocal approximation to obtain a remainder R is discussed in connection with FIG. 6 below.

Multiplying (205) includes obtaining segment-constant C (defined below) for each message segment M_s and multiplying each segment-constant C by its remainder R to obtain a segment-remainder SR for each message segment. Segment-constants C may be obtained based on the position X of message segment M_s in message M modulo generator polynomial P or modulo a field extension of P.

Accumulation (207) includes adding the segment-remainders SR for each message segment M_s to obtain an accumulated-remainder AR for message M.

Moduloing (209) includes dividing accumulated-remainder AR by generator polynomial P, or multiplying AR by a reciprocal approximation of generator polynomial P, to obtain a CRC for message M. However, if the degree of accumulated-remainder AR is less than the degree of polynomial P, moduloing (209) is not needed since the remainder (i.e., the CRC) of message M is accumulated-remainder AR.

FIG. 3 shows an implementation of process 20 for calculating a CRC of message M based on generator polynomial P. For example:

if

M = 10111*000*, (Message 10111 with with 3 zero-bits appended)

s = 2,

n = 4, and

P = 1001 =(Deg(P)=3);

then M may be separated as

M_{s-1} = M_1 = 1011 = segment 33,

$X^{n*(s-1)}$ = X^4 = 10000,

M_{s-2} = M_0 = 1*000* = segment 35,

$X^{n*(s-2)}$ = X^0 = 00001;

where

$$M = 1011*10000 + 1000*00001 = 10111000.$$

CRC generator 30 obtains a CRC for message M based on generator polynomial P, where the CRC for message M is the remainder of message M divided by polynomial P (i.e., CRC = M(modulo)P = MOD(M,P).

Typically, generator polynomials P are selected because they are irreducible (i.e., they have no factors). Several examples of well-known generator polynomials include:

LRCC8	$= X^8 + 1$
	$= 100000001;$
CRC16	$= x^{16} + x^{15} + X^2 + 1$
	$= 11000000000000101;$
SDLC	$= x^{16} + x^{12} + X^5 + 1$
	$= 10001000000100001;$
LRCC	$= x^{16} + 1$
	$= 10000000000000001;$
CRC12	$= x^{12} + x^{11} + X^3 + X^2 + X + 1$
	$= 1100000001111;$ and
ETHERNET	$= x^{32} + x^{26} + X^{23} + X^{22} + x^{16} + x^{11} +$
	$X^{10} + X^8 + x^7 + X^5 + X^4 + X^2 + X + 1$
	$= 100000100110000010000110110110111;$

where LRCC stands for Last Registration Control Channel.

CRC generator 30 includes modulo unit 32, multiplier 34, accumulator 36, and modulo unit 38. Here, modulo unit 32 has modulo units 32a and 32b implemented in hardware.

Modulo unit 32a divides message segment 33 by generator polynomial P to obtain remainder R_{i+1} (i.e., $R_{i+1} = M_{s-1}$(modulo)P = MOD(M_{s-1},P)). Modulo unit 32b divides message segment 35 by

generator polynomial P to obtain remainder R_i (i.e., $R_i = M_{s-2}(\text{modulo})P = \text{MOD}(M_{s-2},P)$). For example:

if

$$\begin{aligned} M &= 10111\mathit{000}, \\ M_{s-1} &= M_1 = 1011 = \text{segment } 33, \\ M_{s-2} &= M_0 = 1000 = \text{segment } 35, \text{ and} \\ P &= 1001; \end{aligned}$$

then

$$\begin{aligned} R_{i+1} = R_1 &= M_{s-1}(\text{modulo})P = \text{MOD}(M_{s-1},P) \\ &= 1011(\text{modulo})1001 = 010, \text{ and} \end{aligned}$$

$$\begin{aligned} R_i &= R_0 = M_{s-2}(\text{modulo})P = \text{MOD}(M_{s-2},P) \\ &= 1000(\text{modulo})1001 = 001; \end{aligned}$$

where

$$\begin{array}{rl} & \underline{0001} \\ P \rightarrow 1001 & \overline{)1011} \leftarrow M_{s-1} \\ & \underline{1001} \\ & \;\,010 = R_{i+1} \end{array} \qquad \begin{array}{rl} & \underline{0001} \\ P \searrow 1001 & \overline{)1000} \leftarrow M_{s-2} \\ & \underline{1001} \\ & \;\,001 = R_i. \end{array}$$

Multiplier 34 multiplies remainders R_{i+1} and R_i by segment-constants C_{i+1} and C_i to obtain segment-remainders SR_{i+1} and SR_i. Here, segment-constants C_{i+1} and C_i are obtained by moduloing the position X of segments 33 and 35 in message M by generator polynomial P (i.e., $C_{i+1} = X^{n*(i+1)}(\text{modulo})P$ and $C_i = X^{n*i}(\text{modulo})P$). For example:

if

$$\begin{aligned} M &= 10111\mathit{000}, \\ P &= 1001, \\ s &= 2, \text{ and} \\ n &= 4; \end{aligned}$$

then for

$$\begin{aligned} SR_{i+1} &= SR_1 \\ C_{i+1} &= C_1 = X^{4*1}(\text{modulo})P \\ &= 10000(\text{modulo})1001 = 010, \text{ and} \end{aligned}$$

$$\begin{aligned} SR_i &= SR_0 \\ C_i &= C_0 = X^{4*0}(\text{modulo})P \\ &= 0001(\text{modulo})1001 = 001; \end{aligned}$$

where

```
                 00010  <-- X^{4*1}        P -->   0000   <-- X^{4*0}
P --> 1001 |10000                         1001 |0001
            1001                                 0000
             0010                                 001 = C_i.
             0000
              010 = C_{i+1}
```

Segment-constants C_{i+1} and C_i may be obtained in advance, based on a known segmentation of message M and stored in memory unit 39, which is accessible to CRC generator 30. In other embodiments, segment-constants C_{i+1} and C_i may be obtained 'on the fly' within CRC generator 30 upon receipt of message M.

Multiplier 34 includes multipliers 34a and 34b. Multiplier 34a multiplies remainder R_{i+1} by segment-constant C_{i+1} to obtain segment-remainder SR_{i+1}. Multiplier 34b multiplies remainder R_i by segment constant C_i to obtain segment-remainder SR_i. For example:

if

$R_{i+1} = 010$, $C_{i+1} = 010$ and (from above)
$R_i = 001$, $C_i = 001$;

then

$SR_{i+1} = R_{i+1}*C_{i+1} = 010*010 = 00100$, and
$SR_i = R_i*C_1 = 001*001 = 00001$;

where

```
       010            001
  *    010       *    001
    + 000           + 001
   + 010           + 000
  + 000           + 000
    00100 = SR_{i+1}  00001 = SR_i.
```

Accumulator 36 adds segment-remainders SR_{i+1} and SR_i together to obtain accumulated-remainder AR. For example:

if

$SR_{i+1} = SR_1 = 00100$, (from above)
$SR_1 = SR_0 = 00001$;

```
then
     AR   = 00100+00001 = 00101,

where
               00100
             + 00001
             -------
               00101 = AR.
```

Modulo unit 38 obtains the CRC for message M by moduloing accumulated-remainder AR by generator polynomial P (i.e., CRC = AR(modulo)P = MOD(AR, P)). For example:

```
if
     AR = 00101, and          (from above)
     P  = 1001;

then
     CRC  = AR(modulo)P = MOD(AR, P)
          = 00101(modulo)1001 = 101,
where
                    00000      AR
     P       1001 | 00101
                    00000
                    -----
                      101 = CRC.
```

Hence, process 20 implemented on CRC generator 30 obtains the same CRC for message M, here 10111*000*. In this example, moduloing AR by polynomial P is not needed since the degree of AR was less than the degree of P.

CRC generator 30 may be expanded to include enough components for obtaining the CRC for message M separated into N segments. FIG. 4 shows CRC generator 40 capable of operating on message M separated into four (4) segments 43, 45, 47 and 49. For example:

```
if
     M = M' = 10111101,   (e.g., message M in
                           the example for FIG.
                           3 above having the
                           obtained CRC appended
                           to it)
     s = 4,
```

$n = 2$, and
$p = 2$;

then M may be separated as

$$
\begin{aligned}
M_{s-1} &= M_3 = 10 = \text{segment } 43,\\
X^{n*(s-1)} &= X^6 = 1000000,\\
M_{s-2} &= M_2 = 11 = \text{segment } 45,\\
X^{n*(s-2)} &= X^4 = 10000,\\
M_{s-3} &= M_1 = \mathit{11} = \text{segment } 47,\\
X^{n*(s-3)} &= X^2 = 100,\\
M_{s-4} &= M_0 = \mathit{01} = \text{segment } 49,\\
X^{n*(s-4)} &= X^0 = 001;
\end{aligned}
$$

where

$$
\begin{aligned}
M = {} & 10*1000000 + 11*10000 +\\
& \mathit{11}*100 + \mathit{01}*001 = 10111\mathit{101}.
\end{aligned}
$$

CRC generator 40 includes modulo unit 42, multipliers 44, accumulator 46, and modulo unit 48. Modulo unit 42 includes modulo units 42a, 42b, 42c and 42d. Modulo units 42a, 42b, 42c and 42d each operate to divide message segment 43, 45, 47 and 49 by generator polynomial P to obtain remainders R_3, R_2, R_1 and R_0. For example:

if

$$
\begin{aligned}
M &= 10111\mathit{101} = M', \qquad \text{(from above)}\\
M_{s-1} &= M_3 = 10 = \text{segment } 43,\\
M_{s-2} &= M_2 = 11 = \text{segment } 45,\\
M_{s-3} &= M_1 = \mathit{11} = \text{segment } 47,\\
M_{s-4} &= M_0 = \mathit{01} = \text{segment } 49, \text{ and}\\
P &= 1001;
\end{aligned}
$$

then

$$
\begin{aligned}
R_{i+3} = R_3 = M_{s-1}(\text{modulo})P &= 10(\text{modulo})1001\\
&= 10,
\end{aligned}
$$

$$
\begin{aligned}
R_{i+2} = R_2 = M_{s-2}(\text{modulo})P &= 11(\text{modulo})1001\\
&= 11,
\end{aligned}
$$

$$
\begin{aligned}
R_{i+1} = R_1 = M_{s-3}(\text{modulo})P &= \mathit{11}(\text{modulo})1001\\
&= 11, \text{ and}
\end{aligned}
$$

$$
R_1 = R_0 = M_{s-4}(\text{modulo})P = \mathit{01}(\text{modulo})1001
$$

= 01.

Multiplier 44 multiplies remainders R_3 to R_0 by segment-constants C_3 to C_0 to obtain segment-remainders SR_3 to SR_0. Segment-constants C_3 to C_0 correspond to each particular segment 43, 45, 47 and 49 and may be obtained by moduloing the position of segments in message M by polynomial P. (i.e., $C_3 = X^{n*(3)}$(modulo)P, $C_2 = X^{n*2}$(modulo)P, $C_1 = X^{n*1}$(modulo)P, $C_0 = X^{n*0}$(modulo)P). For example:

if

M = 10111*101*, (from above)
P = 1001,
s = 4, and
n = 2;

then

$SR_3 = SR_{i+3}$
$C_3 = C_{i+3} = X^{2*3}$(modulo)P
= 1000000(modulo)1001 = 001;

$SR_2 = SR_0$
$C_2 = C_{i+2} = X^{2*2}$(modulo)P
= 10000(modulo)1001 = 010;

$SR_1 = SR_1$
$C_1 = C_{i+1} = X^{2*1}$(modulo)P
= 100(modulo)1001 = 100; and

$SR_0 = SR_i$
$C_0 = C_{i+0} = X^{4*0}$(modulo)P
= 001(modulo)1001 = 001.

Segment constants C_3 to C_0 may be obtained in advance based on the segmentation of message M and stored in a memory unit 39 (FIG. 3) accessible to CRC generator 40. In other embodiments C_3 to C_0 may be obtained 'on the fly' (i.e., in real-time)within CRC generator 40 as it receives message M.

Multiplier 44 multiplies R_3 by C_3, R_2 by C_2, R_1 by C_1, and R_0 by C_0 to obtain segment-remainders SR_3 to SR_0. For example:

if

$$R_{i+3} = R_3 = 10,\quad C_{i+3} = C_3 = 001;\ \text{(from above)}$$
$$R_{i+2} = R_2 = 11,\quad C_{i+2} = C_2 = 010;$$
$$R_{i+1} = R_1 = 11,\quad C_{i+1} = C_1 = 100;\ \text{and}$$
$$R_i = R_0 = 01,\quad C_i = C_{i0} = 001;$$

then

$$SR_3 = R_3{*}C_3 = 10{*}001 = 0010;$$
$$SR_2 = R_2{*}C_2 = 11{*}010 = 0110;$$
$$SR_1 = R_1{*}C_1 = 11{*}100 = 1100;\ \text{and}$$
$$SR_0 = R_0{*}C_0 = 01{*}001 = 0001.$$

Accumulator 46 adds segment-remainders SR_3 to SR_0 together to obtain accumulated-remainder AR. Here, accumulator 46 includes accumulators 46a, 46b and 46c, where accumulators 46a and 46b compute temporary accumulations T_1 and T_0 and accumulator 46c combines temporary accumulations T_1 and T_0 to obtain accumulated-remainder AR. For example:

if

$$SR_{i+3} = SR_3 = 0010,\quad \text{(from above)}$$
$$SR_{i+2} = SR_2 = 0110,$$
$$SR_{i+1} = SR_1 = 1100,\ \text{and}$$
$$SR_i = SR_0 = 0001;$$

then

$$T_1 = 0010+0110 = 0100,$$
$$T_0 = 1100+0001 = 1101,\ \text{and}$$

$$AR = 0100+1101 = 1001.$$

Finally, modulo unit 48 obtains the CRC for message M, here message M' having the CRC obtained as described in FIG. 3 above, by moduloing accumulated-remainder AR by polynomial P (i.e., CRC = AR(modulo)P = MOD(AR, P). For example:

if

$$AR = 1001,\ \text{and}\quad \text{(from above)}$$
$$P = 1001;$$

then

$$CRC = AR(\text{modulo})P = 1001(\text{modulo})1001$$

= 0

where

```
                  00001      AR
    P ---> 1001 |1001   <---
                 1001
                  000 = CRC.
```

Thus, CRC generator 40 verifies the integrity of message M from the example in FIG. 3 where the CRC of message M was appended to form M′ and transmitted to a receiver 14 who confirmed the transmission using CRC generator 40 (FIG. 4).

According to process 20, CRC generators 30 and 40 may be further simplified where the degree of message segments M_s are less than the degree of generator polynomial P (i.e., Deg(M_s) < Deg(P)). As shown in the example for FIG. 4 above, the remainder R of M_s(modulo)P equals M_s when the degree of M_s is less than the degree of P. Thus, CRC generator 50 (FIG. 5) does not need an initial modulo unit (e.g., 32 or 42) for obtaining a remainder R_i of message segments M_s that are of a degree less than the degree of generator polynomial P. For segments of degree equal to P (i.e., Deg(M_s) = Deg(P)) modulo units 32 or 42 may be replaced by an xor, as M_s(modulo)P equals M_s-P.

Here, CRC generator 50 includes multiplier 54, accumulator 56, and modulo unit 58, which operate to obtain a CRC for message M separated into four segments 53, 55, 57 and 59 of a degree less than the degree of generator polynomial P (i.e., Deg(M_s) < Deg(P)). For example:

if

M = 10111*000*, (M including 3 appended zero bits as in FIG. 3 above)

s = 4,
n = 2, and
P = 1001;

then

$$
\begin{aligned}
M_{s-1} &= M_3 = 10 = \text{segment } 53,\\
X^{n*(s-1)} &= X^6 = 1000000,\\
M_{s-2} &= M_2 = 11 = \text{segment } 55,\\
X^{n*(s-2)} &= X^4 = 10000,\\
M_{s-3} &= M_1 = 10 = \text{segment } 57,\\
X^{n*(s-3)} &= X^2 = 100,\\
M_{s-4} &= M_0 = 00 = \text{segment } 59,\\
X^{n*(s-4)} &= X^0 = 001;
\end{aligned}
$$

and

$$
\begin{aligned}
M = {} & 10*1000000 + 11*10000 +\\
& 10*100 + 00*001 = 10111000.
\end{aligned}
$$

Multiplier 54 multiplies segments 53 to 59 by segment-constants C_3 to C_0 to obtain segment-remainders SR_3 to SR_0. Segment-constants C_3 to C_0 may be obtained in advance or calculated 'on the fly' as described above. For example:

if

$$
\begin{aligned}
M &= 10111000, \qquad \text{(from above)}\\
P &= 1001,\\
s &= 4, \text{ and}\\
n &= 2;
\end{aligned}
$$

then

$$
\begin{aligned}
SR_3 &= SR_{i+3}\\
C_3 &= C_{i+3} = X^{2*3}(\text{modulo})P\\
&= 1000000(\text{modulo})1001 = 001;
\end{aligned}
$$

$$
\begin{aligned}
SR_2 &= SR_{i+2}\\
C_2 &= C_{i+2} = X^{2*2}(\text{modulo})P\\
&= 10000(\text{modulo})1001 = 010;
\end{aligned}
$$

$$
\begin{aligned}
SR_1 &= SR_{i+1}\\
C_1 &= C_{i+1} = X^{2*1}(\text{modulo})P\\
&= 100(\text{modulo})1001 = 100; \text{ and}
\end{aligned}
$$

$$
\begin{aligned}
SR_0 &= SR_i\\
C_0 &= C_{i+0} = X^{4*0}(\text{modulo})P\\
&= 001(\text{modulo})1001 = 001.
\end{aligned}
$$

Multiplier 54 multiplies M_3 by C_3, M_2 by C_2, M_1 by C_1, and M_0 by C_0 to obtain segment-remainders SR_3 to SR_0, since each message segment M_s equals its remainder R. For example:

if

$$
\begin{aligned}
M_{s-1} &= M_3 = 10, & C_{1+3} &= C_3 = 001,\\
M_{s-2} &= M_2 = 11, & C_{i+2} &= C_2 = 010,\\
M_{s-3} &= M_1 = 10, & C_{1+1} &= C_1 = 100, \text{ and}\\
M_{s-4} &= M_0 = 00, & C_i &= C_{i0} = 001;
\end{aligned}
$$

then

$$
\begin{aligned}
SR_3 &= M_3{*}C_3 &&= 10{*}001 = 0010,\\
SR_2 &= M_2{*}C_2 &&= 11{*}010 = 0110,\\
SR_1 &= M_1{*}C_1 &&= 10{*}100 = 1000, \text{ and}\\
SR_0 &= M_0{*}C_0 &&= 00{*}001 = 0000.
\end{aligned}
$$

Accumulator 56 adds segment-remainders SR_3 to SR_0 together to obtain accumulated-remainder AR. Here, accumulator 56 includes accumulators 56a, 56b and 56c, where accumulators 56a and 56b compute temporary accumulations T_1 and T_0 and accumulator 56c combines temporary accumulations T_1 and T_0 to obtain accumulated-remainder AR. For example:

if

$$
\begin{aligned}
SR_{i+3} &= SR_3 = 0010, && \text{(from above)}\\
SR_{i+2} &= SR_2 = 0110,\\
SR_{i+1} &= SR_1 = 1000, \text{ and}\\
SR_i &= SR_0 = 0000;
\end{aligned}
$$

then

$$
\begin{aligned}
T_1 &= 0010+0110 = 0100;\\
T_0 &= 1000+0000 = 1000; \text{ and}\\
AR &= 0100+1000 = 1100.
\end{aligned}
$$

Finally, modulo unit 58 obtains a CRC for message M by moduloing accumulated-remainder AR by polynomial P. For example:

if

$$
\begin{aligned}
AR &= 1100, \text{ and} && \text{(from above)}\\
P &= 1001;
\end{aligned}
$$

then

```
CRC  = AR(modulo)P = 1100(modulo)1001
     = 101;
```

where

```
                  0000   AR
           1001 |1100
  P             1001
                 101 = CRC.
```

Thus, CRC generator 50 obtains the same CRC for message M as calculated in the example in FIG. 3 above without needing modulo units 32 or 42 of FIGS 3 and 4.

Moduloing (e.g., (203) and (209)) may also be accomplished by multiplying message M (or message segment M_s) by a reciprocal approximation D of generator polynomial P and subtracting that result from message M (or message segment M_s) to obtain a remainder R. Moduloing by multiplication by reciprocal approximator RA may be obtained based upon the following relationships:

$$RA = X^{p+d}/P;$$

$$M/P = M*RA*1/X^{p+ra} \quad (\text{for } 0<=Deg(M)<= p+ra);$$

$$M = (M/P)*P + M(\text{modulo})P;$$

$$R = M(\text{modulo})P = M - (M/P)*P;$$

$$R = M(\text{modulo})P = M - (M*D/X^{p+ra})*P$$

where X^{p+ra} is a polynomial having a most significant bit of degree p+ra (i.e. $Deg(X^{p+ra})=p+ra$); p is the degree of generator polynomial P (i.e., Deg(P)=p); ra is the degree of reciprocal-approximator RA (i.e., Deg(RA)=ra); and the degree of message M, for which remainder R is sought, is greater than zero and less than or equal to p+ra (i.e., 0 < Deg(M) <= p+ra). For example:

if

```
M = 10111000     (i.e., Deg(M) = 7), and
P = 1001         (i.e., Deg(P) = 3).
```

then reciprocal-approximator RA would have a degree of at least four (4) for p+ra to be greater than or equal to the degree of M, here seven (7). Thus:

```
if
      M = 10111000    (i.e., Deg(M) = 7),
      P = 1001        (i.e., Deg(P) = 3); and;
     ra = 4;

then
      X^(p+ra) = 10000000 (i.e., Deg(X^(p+ra)) = 7), and
      D = X^(p+ra)/P = X^(3+4)/1001
        = 10000000/1001 = 10010;

where
                   00010010 <- RA
              1001|10000000 <-------- X^(p+ra).
       P ->        1001
                    0010
                    0000
                     0100
                     0000
                      1000
                      1001
                       0010
                       0000
                        010
```

Modulo unit 60 may calculate reciprocal-approximator RA prior to receiving message M and store RA in memory 69 since both generator polynomial P and the degree of message M are known prior to receiving message M. In other embodiments, reciprocal-approximator RA may be built in or obtained 'on-the fly' by modulo unit 60 after receiving message M. Once the form of the polynomial is fixed, the implementation of the corresponding hardware may be simplified considerably.

To obtain remainder R for message M modulo unit 60 includes multiplication unit 62, truncation unit 64, multiplication unit 66 and subtraction unit 68 where:

T_0 = M*RA   is performed by unit 62,

$T_1 = T_0/X^{p+ra}$ is performed by unit 64,

$T_2 = T_1 * P$ is performed by unit 66, and

$R = M - T_2$ is performed by unit 68.

Multiplication unit 62 receives message M and multiplies M by reciprocal-approximator RA to obtain temporary result T_0. For example:

if
$M = 10111000$, (from FIG. 3 above)
$P = 1001$, and
$RA = 10010$;

then
$T_0 = M*RA = 10111000*10010$
$= 101011110000$;

where

```
     10111000  <- M
   *    10010  <- RA
   ----------
     00000000
    10111000
   00000000
  00000000
 10111000
 ------------
 101011110000  <- T0.
```

Multiplication unit 62 provides temporary result T_0 to truncation unit 64, which divides T_0 by X^{p+ra}, here 10000000, to obtain truncated result T_1. In other embodiments, truncation unit 64 may remove the p+ra least significant bits of temporary result T_0 without dividing by X^{p+ra} to obtain truncated result T_1. For example:

if
$p = 3$,
$ra = 4$, and
$T_0 = 101011110000$;

then
$p+ra = 7$, and
$T_1 = 10101$.

Thus for p+ra equaling seven (7), the seven (7) least significant bits, here 1110000, are removed from T_0 to obtain T_1.

Truncation unit 64 provides truncated result T_1 to multiplication unit 66, which multiplies T_1 by generator polynomial P to obtain temporary result T_2. For example;

if
P = 1001, and
T_1 = 10101;

then
$T_2 = T_1*P$ = 10101*1001
= 10111101

where

```
  10101  <- T1
*  1001  <- P
  -----
  10101
 00000
00000
10101
--------
10111101  <- T2.
```

Multiplication unit 66 provides temporary result T_2 to subtraction unit 68, which subtracts T_2 from message M to obtain remainder R. For example:

if
M = 10111000, and (from above)
T_2 = 10111101;

then
R = M - T_2 = 101

where

```
   10111000  <- M
 - 10111101  <- T2
   --------
   00000101  <- R.
```

Thus, modulo unit 60 obtains remainder R for message M using multiplication by reciprocal approximation. Hence, modulo unit 60 may calculate the CRC for the entire message M on its

own, or may be incorporated into CRC generators 30 and 40 to obtain remainders R for message segments M_s.

For example, FIG. 7 shows an implementation of the CRC generator in FIG. 3 employing modulo unit 60 in FIG. 6. Here, modulo units 60 are show as MH(M, RA, P). For example:

if

$M = 10111000$, (Same as in FIG. 3 above)

$s = 2$,

$n = 4$, and

$P = 1001$;

then M may be separated as

$M_{s-1} = M_1 = 1011$ = segment 73

$X^{n*(s-1)} = X^4 = 10000$,

$M_{s-2} = M_0 = 1000$ = segment 75

$X^{n*(s-2)} = X^0 = 00001$;

where

$M = 1011*10000 + 1000*00001 = 10111000$.

CRC generator 70 obtains a CRC for message M based on generator polynomial P, where the CRC for message M is the remainder of message M divided by polynomial P.

CRC generator 70 includes modulo unit 72, multiplier 74, accumulator 76, and modulo unit 78. Here, modulo unit 72 includes modulo units 72a and 72b, which multiply message segments 73 and 75 by a reciprocal approximation of generator polynomial P to obtain remainders R_{i+1} and R_i.

Modulo unit 72a multiplies message segment 73 by reciprocal-approximator RA of generator polynomial P to obtain a remainder R as shown in FIG. 6. For example:

if

$M = 10111000$,

$M_{s-1} = M_1 = 1011$ = segment 73,

$M_{s-2} = M_0 = 1000$ = segment 75,

$\mathrm{Deg}(M_{s-1}) = 3$

$\mathrm{Deg}(M_{s-2}) = 3$

$P = 1001$, and
$RA = X^{p+ra}/P = X^{3+1}/P$, so that p+ra is greater than or equal to the degree of each message segment M_{s-1} and M_{s-2};

then

$$RA = X^{3+1}/P = 10000/1001 = 10;$$

where

```
                  00010 <-- RA
   P -->    1001|10000 <-- X^{3+1}
                 1001
                  0000
                  0000
                   010;
```

and

$$T_{0(i+1)} = M_{s-1}*RA = 1011 * 10 = 10110,$$
$$T_{0(i)} = M_{s-2}*RA = 1000 * 10 = 10000,$$

$$T_{1(i+1)} = T_{0(i+1)}/X^{3+1} = 10110/10000 = 1,$$
$$T_{1(i)} = T_{0(i)}/X^{3+1} = 10000/10000 = 1,$$

$$T_{2(i+1)} = T_{1(i+1)}*P = 1*1001 = 1001,$$
$$T_{2(i)} = T_{1(i)}*P = 1*1001 = 1001,$$

$$R_{i+1} = M_{s-1} - T_{2(i+1)} = 1011 - 1001 = 010$$
$$R_i = M_{s-2} - T_{2(i)} = 1000 - 1001 = 001.$$

Hence, modulo units 72a and 72b obtain the same remainders R_{i+1} and R_i as modulo units 32a and 32b in FIG. 3 above.

Multiplier 34 multiplies R_{i+1} and R_i by segment-constants C_{i+1} and C_i to obtain segment-remainders SR_{i+1} and SR_i. Here, segment-constants C_{i+1} and C_i are obtained 'on the fly' by moduloing the position X of segments 33 and 35 in message M by generator polynomial P (i.e., $C_{i+1} = X^{n*(i+1)}$(modulo)P and $C_i = X^{n*i}$(modulo)P) using modulo unit 60 described in FIG. 6. For example:

if

$$X^{n*(i+1)} = X^{4*(1)} = M_1 = 10000,$$
$$X^{n*1} = X^{4*(0)} = M_0 = 00001,$$
$$Deg(X^{4*(1)}) = 4,$$

$Deg(X^{4*(0)}) = 0$,
$P = 1001$, and
$RA = X^{p+ra}/P = X^{3+1}/P$, so that p+ra is greater than or equal to the degree of each message segment $X^{4*(1)}$ and $X^{4*(0)}$;

then

$$RA = 10000/1001 = 10;$$

and

$$T_{0(i+1)} = M_1 * RA = 10000 * 10 = 100000,$$
$$T_{0(i)} = M_0 * RA = 00001 * 10 = 000010,$$

$$T_{1(i+1)} = T_{0(i+1)}/X^{3+1} = 100000/10000 = 10,$$
$$T_{1(i)} = T_{0(i)}/X^{3+1} = 000010/10000 = 0,$$

$$T_{2(i+1)} = T_{1(i+1)} * P = 10 * 1001 = 10010,$$
$$T_{2(i)} = T_{1(i)} * P = 0 * 1001 = 00000,$$

$$C_{i+1} = M_1 - T_{2(i+1)} = 10000 - 10010 = 010,$$
$$C_i = M_0 - T_{2(i)} = 00001 - 00000 = 001.$$

In other embodiments segment-constants C_{i+1} and C_i may be obtained in advance in stored in a memory unit (e.g. 39).

Multiplier 74 includes multipliers 74a and 74b. Multiplier 74a multiplies remainder R_{i+1} by segment-constant C_{i+1} to obtain segment-remainder SR_{i+1}. Multiplier 74b multiplies R_i by segment constant C_i to obtain segment-remainder SR_i. For example:

if
$R_{i+1} = 010$, $C_{i+1} = 010$ and (from above)
$R_i = 001$, $C_i = 001$;

then
$$SR_{i+1} = R_{i+1} * C_{i+1} = 010 * 010 = 00100, \text{ and}$$
$$SR_i = R_i * C_i = 001 * 001 = 00001;$$

Accumulator 76 adds segment-remainders SR_{i+1} and SR_i together to obtain accumulated-remainder AR. For example:

if
$$SR_{i+1} = SR_1 = 00100,$$
$$SR_i = SR_0 = 00001;$$

then
$$AR = 00100+00001 = 00101.$$

Modulo unit 78 obtains a CRC for message M by moduloing accumulated-remainder AR by generator polynomial P. Here, modulo unit 78 obtains the CRC by using multiplication by reciprocal approximation shown in FIG. 6. For example:

if

AR = M = 00101,
Deg(AR) = 2
P = 1001, and
RA = $X^{p+ra}/P = X^{3+1}/P$ so that p+ra is greater than or equal to the degree of the message for which a remainder is desired, here AR;

then

$$RA = 10000/1001 = 10;$$

and

$$T_0 = M*RA = 00101*10 = 1010,$$

$$T_1 = T_0/X^{3+1} = 1010/10000 = 0,$$

$$T_2 = T_1*P = 0*1001 = 0,$$

$$R = CRC = M - T_2 = 00101 - 0 = 101.$$

Thus CRC generator 70 obtains the same CRC the example for CRC generator 30. Likewise, CRC generator 70 may also be expanded to include enough components for obtaining the CRC for message M separated into N segments.

CRC generator 80 (FIG. 8) includes subtraction unit 82, modulo unit 84 and accumulator 86 for updating a CRC of a

message M adjusted during transmission. Subtraction unit 82 subtracts old message 83 from new message 85 to obtain difference D. For example:

if

$P = 1001,$ $\quad$ ← CRC_{old}

$M_{old} = 10111|101$

$M_{new} = 10001|101$ $\quad$ ← CRC to be updated

then

$D = M_{new} - M_{old} = 00110000$

wherein $\quad$ adjusted portion of M

```
 10|11|1101 ←  Mnew
-10|00|1101 ←  Mold
 00|11|0000 = D.
```

Modulo unit 84 modulos difference D by generator polynomial P to obtain a difference-remainder DR. For example:

if

$P = 1001,$ and

$D = 00110000;$

then

$DR = D(\text{modulo})P = MOD(D,P) =$

wherein

```
          00000110
     1001|00110000     ← D
P →       0000
           0110
           0000
            1100
            1001
             1010
             1001
              0110
              0000
               110 = DR.
```

In other embodiments, difference-remainder DR may be obtained using multiplication by reciprocal-approximator RA (i.e. MH(D,RA,P)).

Accumulator 86 adds difference-remainder DR and CRC_{old} to obtain a CRC_{new}. For example:

if

$$CRC_{old} = 101 \text{ and}$$
$$DR = 110;$$

then

$$CRC_{new} = CRC_{old} + DR = 101+110$$
$$= 011;$$

where

$$\begin{array}{r} 101 \\ +\ 110 \\ \hline 011 \end{array} = CRC_{new}.$$

The accuracy of this CRC_{new} may be confirmed by replacing CRC_{old} in the adjusted message M_{new} with CRC_{new} and determining whether M_{new}(modulo)CRC_{new} equals zero. For example:

if

$$P = 1001,$$
$$M_{new} = 10001|\underbrace{011}_{CRC_{new}};$$

then

$$M_{new}(\text{modulo})CRC_{new} = 0$$

wherein

```
         00010011
    1001|10001011
         1001
          0011
          0000
           0110
           0000
            1101
            1001
             1001
             1001
```

$$000 = CRC.$$

CRC generator 90 (FIG. 9) includes subtraction unit 92, modulo unit 94, multiplier 96, modulo unit 98 and accumulator 99 for updating a CRC of a message M adjusted during transmission. CRC generator 90 differs from generator 80 in that it adjusts the CRC of a message M based on the adjusted segment of the message.

Subtraction unit 92 subtracts old message segment 93 from new message segment 95 to obtain difference-segment DS. For example:

if

$$\begin{aligned}
P &= 1001,\\
n &= 2\\
s &= 4\\[1em]
M_{old} &= 10111|101 \longleftarrow CRC_{old}\\
M_{s-1} &= 10\\
M_{s-2} &= 11 = \text{segment } 93\\
M_{s-3} &= 11\\
M_{s-4} &= 01\\[1em]
M_{new} &= 10001|101\\
M_{s-1} &= 10\\
M_{s-2} &= 00 = \text{segment } 95\\
M_{s-3} &= 11\\
M_{s-4} &= 01
\end{aligned}$$

then

$$DS = M_{s-2(new)} - M_{s-2(old)} = 00 - 11 = 11.$$

Modulo unit 94 modulos difference-segment DS by generator polynomial P to obtain a difference-segment-remainder DSR. For example:

if

$$\begin{aligned}
P &= 1001, \text{ and}\\
DS &= 11;
\end{aligned}$$

then

$DSR = DS(modulo)P = MOD(DS,P) = 11$

wherein

```
              00
        1001 |11  <-- DS
P -->         00
              11 = DSR.
```

Here, as above, if the difference-segment DS is of a lesser degree than polynomial P, modulo unit 94 is not needed since the modulo of DS equals DS.

Multiplier 96 multiplies difference-segment-remainder DSR by an appropriate segment-constant C_i to obtain an expanded segment-remainder ESR. Segment-constants C_3 to C_0 for this example may be obtained as described above. For example:

if

$DSR = (M_{2new} - M_{2old})(modulo)P = 11$

and

$$\begin{aligned} C_i &= C_2 = X^{2*2}(modulo)P \\ &= 10000(modulo)1001 = 010; \end{aligned}$$

then

$EDR = DSR * C_i = 11*010 = 110.$

Modulo unit 98 obtains message difference-remainder DR by moduloing the extended difference-remainder by generator polynomial P. For example:

if

$P = 1001$ and

$EDR = 110;$

then

$DR = 110.$

Again, for extended difference-remainders of a degree less than the degree of polynomial P the DR is the EDR.

Finally, accumulator 99 adds the message difference-remainder DR and CRC_{old} to obtain a CRC_{new}. For example:

if

$$CRC_{old} = 101 \text{ and}$$
$$DR = 110;$$

then

$$CRC_{new} = CRC_{old} + DR = 101 + 110 = 011;$$

wherein

$$\begin{array}{r} 101 \\ +\ 110 \\ \hline 011 \end{array} = CRC_{new}.$$

All of the above algorithms may be affected by embedding generator polynomial P in a larger ring. For example, let

$$F = P*Q;$$

where F is a field extension of P, Q is an extender, and the greatest common denominator between P and Q is one (1). Segment-constants C may now be calculated using field extension F, instead of p, and message segments Ms increased in size (by bit) accordingly without requiring the additional modulos 42 and 42 in FIG. 3 and 4 above. Rather, only modulo by P, as shown in FIG.5 may be needed.

FIG. 10 shows a general-purpose computer 100 for obtaining a CRC using process 20 or any of the operations of the CRC generator units 30, 40, 50, 60, 70, 80 and 90 shown above. Computer 100 includes a processor 102 (e.g. a CPU), a storage medium 104 (e.g., a random access memory) and communication interface 106 (e.g., a network card) having one or more external connections 106a, 106b and 106c for sending and receiving data transmissions. Storage medium 104 stores computer instructions 108 for obtaining a CRC via process 20 or the operations of the

CRC generator units described above. In one embodiment, computer 100 obtains a CRC for a message M based on multiplication by reciprocal approximation.

Process 20 and the operations of the CRC generators shown above, however, are not limited to use with any particular hardware or software configuration; they may find compatibility in any computing or processing environment. Process 20 may be implemented in hardware, software, or any combination of the two. So too, may the operations of the CRC generator units 30, 40, 50, 60, 70, 80 and 90.

Process 20 and the CRC generators described above may be implemented in computer programs executing on programmable computers that each include a processor, a storage medium readable by the processor (e.g. volatile memory, non-volatile memory, etc.), one or more input devices, and one or more out devices. Program code may be applied to data entered using an input device to perform process 20 or any of the operations of the CRC generators described above. The output information may be applied to one or more output devices, such as screen 110.

Each such program may be implemented in a high level procedural or object-oriented programming language to communicate with a computer system. However, the programs can be implemented in assembly or machine language. The language may be a compiled or an interpreted language.

Each computer program may be stored on an article of manufacture, such as a CD-ROM, hard disk, or magnetic diskette, that is readable by computer 100 to obtain a CRC for message M in the manners described above. Process 20 and the operations for implementing the CRC generators above may also be implemented as a machine-readable storage medium, configured with one or more computer programs, where, upon execution,

instructions in the computer program(s) cause the processor 102 to operate as described above.

A number of embodiments of the invention have been described. Nevertheless, it will be understood that various modifications may be made without departing from the spirit and scope of the invention. For example, message M may be divided into an odd number of segments or segment sizes or field extensions F may be substituted for generator polynomial P were appropriate. Accordingly, other embodiments are within the scope of the following claims.